UNIVERSIDADE FEDERAL DA PARAÍBA

CIÊNCIA DA COMPUTAÇÃO

ALINE MOURA ARAÚJO

CLASSIFICAÇÃO E DETECÇÃO DE PESSOAS EM AMBIENTES NÃO CONTROLADOS UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

JOÃO PESSOA

2019

ALINE MOURA ARAÚJO

CLASSIFICAÇÃO E DETECÇÃO DE PESSOAS EM AMBIENTES NÃO CONTROLADOS UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

Monografia apresentada ao curso Ciência da Computação do Centro de Informática, da Universidade Federal da Paraíba, como requisito para a obtenção do grau de Bacharel em Ciência da Computação.

JOÃO PESSOA

2019

Catalogação na publicação
Seção de Catalogação e Classificação

A663c Araujo, Aline Moura.
CLASSIFICAÇÃO E DETECÇÃO DE PESSOAS EM AMBIENTES NÃO CONTROLADOS UTILIZANDO REDES NEURAIS CONVOLUCIONAIS / Aline Moura Araujo. - João Pessoa, 2019.
45 f. : il.

Orientação: Leonardo Vidal Batista.
Monografia (Graduação) - UFPB/CI.

1. Aprendizado Profundo. 2. Redes Neurais Convolucionais. 3. Classificação de faces. 4. Detecção de faces. I. Batista, Leonardo Vidal. II. Título.

UFPB/CI

CENTRO DE INFORMÁTICA

UNIVERSIDADE FEDERAL DA PARAÍBA

Trabalho de Conclusão de Curso de Ciência da Computação intitulado CLASSIFICAÇÃO E DETECÇÃO DE PESSOAS EM AMBIENTES NÃO CONTROLADOS UTILIZANDO REDES NEURAIS CONVOLUCIONAIS de autoria de Aline Moura Araújo, aprovada pela banca examinadora constituída pelos seguintes membros:

Prof. Dr. Leonardo Vidal Batista
CI/UFPB

Dr. João Janduy Brasileiro Primo
Vsoft Tecnologia

Bel. Paulo Ricardo Pereira Silva
Indra Company

Prof. Dr. Gustavo H. M. B. Motta
Coordenador Bacharelado
Ciência da Computação
CI-UFPB / Mat. SIAPE 2126491

João Pessoa, 27 de Setembro de 2019

Centro de Informática, Universidade Federal da Paraíba
Rua dos Escoteiros, Mangabeira VII, João Pessoa, Paraíba, Brasil CEP: 58058-600
Fone: +55 (83) 3216 7093 / Fax: +55 (83) 3216 7117

“Que importa que ao chegar, eu nem pareça pássaro. Que importa que ao chegar eu venha me arrebentando, Caindo aos pedaços, Sem aprumo e sem beleza. Fundamental é cumprir a missão E cumpri-la até o fim".

(D. Helder Camara)

AGRADECIMENTOS

Agradeço primeiramente à Deus, que em sua infinita sabedoria colocou força em meu coração para vencer essa etapa de minha vida.

Sou grata ao meu orientador e tutor do PET Computação UFPB, Leonardo Vidal Batista, por toda experiência e conhecimento agregados. À todos os amigos que fiz ao longo do caminho e que de alguma forma me ajudaram nessa trajetória, especialmente à Luiz Henrique, namorado querido, que muitas vezes abriu mão dos seus compromissos para ficar ao meu lado. Agradeço também à Paulo Ricardo, pela ajuda imprescindível no desenvolvimento deste trabalho.

Dedico esta monografia à toda minha família e à minha tia Paula Moura (in memoriam) que não pôde estar ao meu lado neste momento tão importante, mas que sempre torceu muito por mim. E, especialmente aos meus pais, pelo carinho, afeto, dedicação e cuidado que me deram durante toda a minha existência e que tornou esse momento possível.

RESUMO

A detecção e contagem de pessoas em ambientes complexos constituem um desafio na área da Visão Computacional. Esse problema se torna ainda mais difícil quando se trata de experimentos em ambientes não controlados, apresentando diversos obstáculos que influenciam de maneira substancial na acurácia do processo. Oclusão total ou parcial, não uniformidade de iluminação, ruído, variação de pose, variação de escala, são alguns dos problemas mais comuns a serem tratados. Uma das formas utilizadas para amenizar a influência desses fatores é treinando os algoritmos de detecção utilizando o maior número de exemplos rotulados possível, nas várias situações que podem ocorrer em situações cotidianas. Adicionalmente, podem-se aplicar métodos de pré-processamento de imagens para mitigar alguns dos obstáculos citados, notadamente a não uniformidade de iluminação. O presente trabalho propõe investigar soluções, utilizando Redes Neurais Convolucionais (CNN), para a detecção e classificação de faces em ambientes complexos e não controlados. Foram avaliadas variantes de uma arquitetura de CNN, uma clássica, denominada LeNet, bem como a aplicação de pré-processamento CLAHE nas imagens de interesse. Nos testes empíricos, os modelos desenvolvidos obtiveram uma acurácia média acima de 98%.

Palavras-chave: Aprendizado Profundo, Redes Neurais Convolucionais, Classificação de faces, Detecção de faces

ABSTRACT

Detecting and counting people in complex environments is a challenge in Computer Vision. This problem becomes even more difficult when it comes to experiments in uncontrolled environments, presenting several obstacles that substantially influence the accuracy of the process. Total or partial occlusion, non-uniformity of illumination, noise, pose variation, scale variation are some of the most common problems to be addressed. One of the ways used to mitigate the influence of these factors is by training detection algorithms using as many labeled examples as possible in the various situations that can occur in everyday situations. Additionally, image preprocessing methods may be applied to mitigate some of the obstacles mentioned, notably the non-uniformity of illumination. The present work proposes to investigate solutions using Convolutional Neural Networks (CNN) for the detection and classification of faces in complex and uncontrolled environments. Variants of a classical CNN architecture called LeNet were evaluated, as well as the application of CLAHE preprocessing to the images of interest. In the empirical tests, the developed models obtained an average accuracy above 98%.

Keywords: Deep Learning, Convolutional Neural Networks, Face Classification, Face Detection

LISTA DE FIGURAS

Figura 1 - Modelo de um Neurônio Artificial. Fonte: Adaptado de HAYKIN, Simon.

Figura 2 - Exemplo de Funções de Ativação. Fonte: FERREIRA, Arthur.

Figura 3 - Problema de classificação binária linearmente separável. Fonte: Medium.com.

Figura 4 - Rede Neural Multicamadas Genérica. Fonte: Adaptado de Medium.com.

Figura 5 - Exemplo de Aplicação da Função Softmax. Fonte: Sefiks.com.

Figura 6 - Exemplo de um codificador One-hot. Fonte: Elaborado pelo autor.

Figura 7 - Operação de Convolução. Fonte: Adaptado de http://deeplearning.stanford.edu/.

Figura 8 - Exemplo de convolução com uso de *strides*. Fonte: Elaborado pelo autor.

Figura 9 - Exemplo de utilização de padding. Fonte: Elaborado pelo autor.

Figura 10 - *Max Pooling* com um filtro de tamanho 2x2 e *stride* 2. Fonte: HIJAZI, Samer.

Figura 11 - Arquitetura original da LeNet-5. Fonte: LECUN, Yann.

Figura 12 - Desempenho da rede.

LISTA DE ABREVIATURAS

CNN - Redes Neurais Convolucionais (*Convolutional Neural Network*)

DL - Aprendizado Profundo (*Deep Learning*)

FCL - Camada Totalmente Conectada (*Fully Connected Layer*)

IA - Inteligência Artificial

ML - Aprendizado de Máquinas (*Machine Learning*)

MLP - Perceptron Multicamadas (*Multilayer Perceptron*)

PIL - Biblioteca de Manipulação de Imagens (*Python Imaging Library*)

RNA - Redes Neurais Artificiais

SGD - Gradiente Descendente Estocástico (*Stochastic Gradient Descent*)

UFPB - Universidade Federal da Paraíba

LISTA DE TABELAS

Tabela 1 - Exemplificação de uma matriz de confusão.

Tabela 2 - Codificação One-hot das classes.

Tabela 3 - Sumário da arquitetura da rede.

Tabela 4 - Parâmetros de inicialização.

Tabela 5 - Desempenho da rede.

Tabela 6 - Matriz de confusão referente à classificação.

Tabela 7 - Tempo de treinamento e teste da rede.

SUMÁRIO

1 INTRODUÇÃO **14**

1.1 Definição do Problema 15

1.2 Objetivos Gerais 15

1.3 Objetivos Específicos 15

2 APRENDIZADO DE MÁQUINA **16**

2.1 Redes Neurais Artificiais **16**

2.1.1 Função de Ativação 18

2.1.1.1 Sigmóide 19

2.1.1.2 Tangente Hiperbólica 19

2.1.1.3 ReLU 20

2.1.2 Rede Neural Multicamadas 20

2.1.3 Função Softmax 22

2.1.4 Codificação One-Hot 23

2.1.5 Função de Custo 24

2.1.6 Cross Entropy 24

2.1.7 Treinamento de uma Rede Neural Multicamadas 25

2.2 Aprendizado Profundo 25

2.2.1 Função de Otimização 26

2.2.1.1 Adadelta 27

2.2.2 Normalização de Batch 27

2.3 Rede Neural Convolucional 27

2.3.1 Camada de Convolução 28

2.3.4 Camada de Pooling 30

2.3.5 Camada Totalmente Conectada (Fully Connected Layer) 31

3 METODOLOGIA **32**

3.1 Classificação 32

3.1.1 Material 32

3.1.1.1 Keras 32

3.1.1.2 Base de Dados 32

3.1.2 Métricas 33
3.1.3 Pré Processamento 34
3.1.3.1 Imagem 34
3.1.3.2 Equalização de Histograma 34
3.1.3.3 Codificação One-hot 35
3.1.4 Treinamento da rede 35
3.1.4.1 Conjunto de Dados de Treinamento, Validação e Teste 35
3.1.4.2 Arquitetura 36
3.2 Detecção 37
4 RESULTADOS 38
4.1 Classificação 38
4.2 Detecção 42
5 CONCLUSÃO E TRABALHOS FUTUROS 42
REFERÊNCIAS 43

1 INTRODUÇÃO

O problema de detecção e contagem de pessoas em ambientes complexos e não controlados têm sido extensivamente investigado no âmbito da Visão Computacional [1][2]. As aplicações práticas são diversas: avaliação do número de pessoas em grandes manifestações públicas e em espetáculos; monitoramento de frequência em salas de aula e em ambientes de trabalho; fluxo de clientes em corredores de lojas de departamento ou centros comerciais; fluxo de passageiros em aeroportos e estações ferroviárias, rodoviárias e metroviárias, fluxo de pedestres em vias públicas; e contagem de pessoas em parques e clubes.

A informação sobre o número de pessoas em um determinado ambiente pode ajudar em vários cenários, de decisões de *marketing* à alocação de recursos para melhoria da segurança pública.

Embora métodos eficazes para contagem de pessoas tenham sido desenvolvidos [3][4], ainda é uma tarefa desafiadora, especialmente quando se trata de ambientes complexos. De fato, há muitas dificuldades na implementação de um sistema de contagem de pessoas aplicável, como tratamento de sombras e oclusão, correção de iluminação deficiente ou não homogênea, e imagens de má qualidade.

Considera-se não controlado qualquer ambiente em que a captura das imagens não obedeça a controle rígido de posicionamento e ajustes de câmeras e dispositivos de iluminação, nem imponha ao indivíduo restrições de pose e colocação em relação a outros objetos e indivíduos. O problema torna-se então mais desafiador, pelo motivos citados anteriormente, alguns dos quais são discutidos a seguir.

Ultimamente, os métodos clássicos de reconhecimento de padrões tem sido progressivamente substituídos por técnicas de Aprendizagem Profunda (*Deep Learning*, DL), que têm alcançado resultados superiores. Não há um consenso sobre a distinção precisa entre

redes profundas e rasas. Porém, as redes mais profundas, de forma geral, conseguem extrair melhores características do que técnicas clássicas quando se tem uma base dados grande.

1.1 Definição do Problema

Apesar a detecção de pessoas por sistemas de visão computacional ainda ser um desafio, muitas metodologias para melhorar o desempenho dos algoritmos já existentes vêm sendo propostas, principalmente pela aplicabilidade na automatização de detecção de pessoas em ambientes como shoppings, locais de trabalho e salas de aulas.

Por uma determinação do Conselho Nacional de Trânsito (CONTRAN), com o intuito de impedir fraudes e para facilitar o monitoramento da frequência de alunos em salas de aula nas autoescolas, abordagens de aprendizagem de máquina vem sendo aplicadas no desenvolvimento de um sistema que seja capaz de detectar e reconhecer os alunos dentro da sala de aula.

1.2 Objetivos Gerais

Este trabalho tem como objetivo geral pesquisar e desenvolver métodos de processamento de imagem capazes de reduzir o tempo de treinamento e melhorar a acurácia de redes neurais convolucionais (*Convolutional Neural Networks*, CNN) para detecção de pessoas em uma foto ou um quadro de um vídeo.

1.3 Objetivos Específicos

Como objetivos específicos, pretende-se:

- Implantar um ambiente de programação adequado para implementação, treinamento e testes de arquiteturas selecionadas, usando os ambientes de desenvolvimento existentes na comunidade (código aberto);
- Investigar arquiteturas avançadas de redes neurais convolucionais;
- Realizar o treinamento de uma rede neural convolucional para a classificação e detecção de pessoas.

2 APRENDIZADO DE MÁQUINA

O aprendizado de máquinas (do inglês, *Machine Learning* - ML) pode ser definido como um subconjunto da inteligência artificial (IA), no qual o estudo científico de algoritmos e modelos estatísticos que os sistemas computacionais usam para executar uma tarefa específica sem usar instruções explícitas, confiando em padrões e inferência, dessa forma, pode-se afirmar que o ML é um método de análise de dados que automatiza a construção de modelos analíticos. Os algoritmos de ML constroem um modelo matemático baseado em dados de amostra, conhecidos como "dados de treinamento", para fazer predições ou decisões [5].

Esse ramo da IA se baseia na ideia de que sistemas podem aprender com dados, identificar padrões e tomar decisões com o mínimo de intervenção humana.

Algoritmos de ML possuem grande espectro de aplicações, como filtragem de e-mail e visão computacional, onde é difícil ou inviável desenvolver um algoritmo convencional para executar a tarefa com eficiência.

Este capítulo tem como objetivo compreender algumas áreas relacionadas à ML utilizadas neste trabalho. Iniciando sobre uma breve introdução sobre Redes Neurais Artificiais (RNAs), explanando algumas técnicas e termos, tais como, função *Softmax*, codificação *one-hot* e *Cross entropy*. Em seguida, será apresentado os fundamentos teóricos de Deep Learning (DL) e CNNs, com os principais métodos utilizados e seus benefícios.

2.1 Redes Neurais Artificiais

O desenvolvimento de uma rede neural artificial tem uma história interessante. Na década de 1940, cientistas descobriram que a fisiologia do cérebro era similar ao modo de processamento utilizados pelos computadores. Uma grande quantidade de dados era utilizada em ambos os casos. No cérebro, o elemento de processamento é o neurônio, enquanto que nos computadores, o elemento de processamento são os *bits*.

Em 1943, McCulloch e Pitts [6] conceberam o primeiro modelo formal de um neurônio de computação elementar. Seu modelo de neurônio fomentou as bases para o desenvolvimento futuro. Em 1949, Hebb [7] estabeleceu uma regra de aprendizado que se

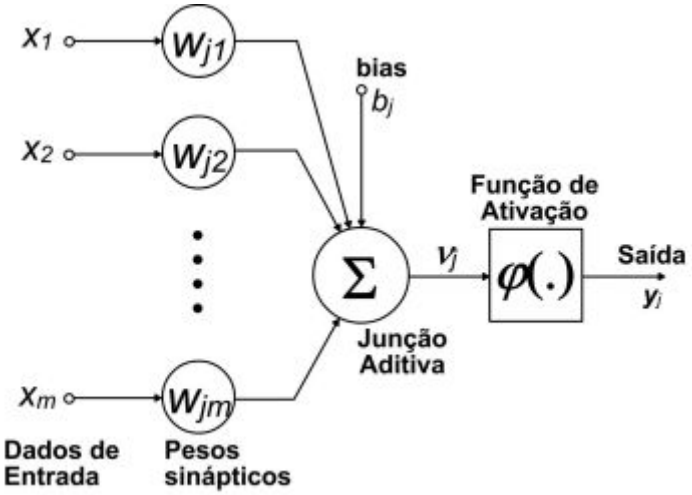
tornou uma contribuição primordial para o desenvolvimento das RNAs que conhecemos hoje. Hebb afirmou que a informação pode ser armazenada em conexões e propôs um esquema de aprendizado para atualizar as conexões de um neurônio.

Durante a década de 1950, o primeiro neurocomputador foi construído e testado. Muitas implementações de computadores neurais foram realizadas na década de 1960, no entanto, como os recursos computacionais disponíveis na época eram relativamente limitados, os teoremas existentes naquele tempo não suportavam problemas computacionais mais complexos. Como consequência dessa limitação de recursos, a pesquisa em redes neurais entrou em uma fase de estagnação e foi só em meados da década de 1980 que as pesquisas em redes neurais foram retomadas.

Atualmente, as RNAs são muito utilizadas em tarefas de reconhecimento de padrões, tais como, reconhecimento de fala e objetos, identificação de células cancerígenas, entre outros [8].

Como dito anteriormente, assim como uma rede biológica natural, as unidades básicas da rede neural artificial são os neurônios. O neurônio artificial apresentado na Figura 1 é um modelo genérico de um neurônio.

Figura 1 - Modelo de um Neurônio Artificial.



Fonte: Adaptado de HAYKIN, Simon.

Cada elemento desse neurônio possui as seguintes funções:

1) Dados de entrada (X_m): os sinais de entrada são conjuntos de dados fornecidos que servirão para o treinamento da rede.

2) Pesos Sinápticos (W_{jm}): cada sinal de entrada é associado à um peso diferente. Dessa forma, os pesos são responsáveis por designar a importância de cada entrada para a saída desse neurônio ao longo do treinamento.

3) Limiar de ativação ou *bias* (b_j): parâmetro que auxilia na adaptação da rede.

4) Junção Aditiva ou Somatório ($\sum$): realiza um somatório ponderado dos sinais de entrada (que são os dados de entrada multiplicados por seus pesos).

5) Função de ativação ($\Phi(.)$): cada neurônio aplica uma função de ativação, geralmente não-linear, à sua entrada de rede para determinar sua saída.

6) Saída (y_j): saída do neurônio com o seu resultado estimado.

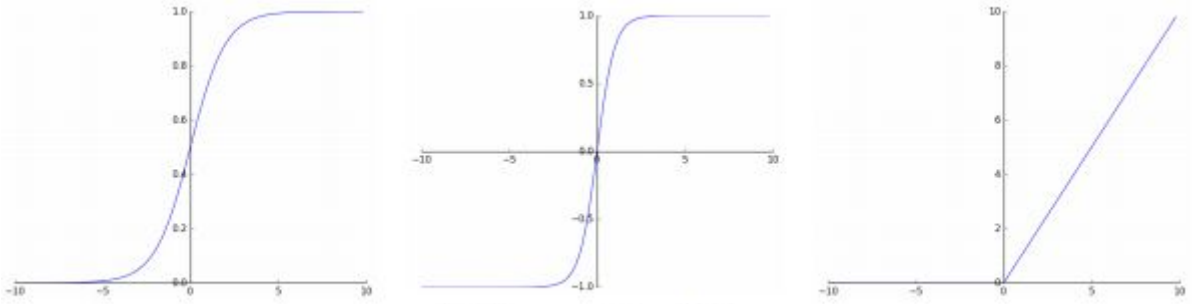
A performance desse neurônio pode ser descrita matematicamente através da equação 1 apresentada abaixo.

$$y_j = \varphi\left(\sum_{i=1}^{m} x_i w_{ji} + b_j\right) \tag{1}$$

2.1.1 Função de Ativação

As funções de ativação utilizadas neste trabalho são funções não-lineares. Essas funções estão associadas ao final da estrutura do neurônio, como foi possível visualizar na Figura 1. Essas funções definem a saída da rede baseadas na entrada e no limiar de ativação fornecidos. As funções de ativação mais comuns são as demonstradas na Figura 2.

Figura 2 - Exemplo de Funções de Ativação.



(a) Sigmóide (b) Tangente Hiperbólica (c) Função Linear Retificada

Fonte: FERREIRA, Arthur.

2.1.1.1 Sigmóide

A principal razão pela qual usamos a função sigmóide é porque ela existe entre os valores (0 a 1). Portanto, essa função é especialmente usado para modelos em que se deve prever a probabilidade como uma saída. Como a probabilidade de algo existe apenas entre o intervalo de 0 e 1, sigmóide seria a escolha mais adequada.

A curva da função sigmóide, Figura 2 (a), tem um formato de 'S' e pode ser representada matematicamente pela equação 2.

$$\varphi(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

2.1.1.2 Tangente Hiperbólica

A função de ativação Tangente Hiperbólica (Tanh) é parecida com a função Sigmóide, porém ela é simétrica em relação à origem, por isso ela varia entre os valores (-1 a 1). A função de ativação Tanh pode ser representada matematicamente pela equação 3 ou pela equação 4.

$$tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})} \tag{3}$$

$$tanh(x) = 2 * \varphi(2x) - 1 \quad (4)$$

2.1.1.3 ReLU

A função Linear Retificada (*Rectified Linear Unit*, ReLU), foi introduzida pela primeira vez por Hahnloser et al.em 2000, em uma rede dinâmica. Em 2011, o uso da ReLU como uma não-linearidade foi mostrado para permitir o treinamento de redes neurais supervisionadas profundas sem a necessidade de pré-treinamento não supervisionado [10].

A ReLU, comparada às funções sigmóide ou Tanh, como mencionadas acima, permitem um treinamento mais rápido e efetivo de arquiteturas mais profundas em conjuntos de dados grandes e complexos, pois, diferente das demais, a ReLU não faz uso de expoentes, ela realiza apenas operações de comparação, adição e multiplicação.

A ReLU é uma função de ativação definida apenas como a parte positiva de seu argumento, ou seja, sempre que houver valores negativos na sua entrada, esse valor será convertido para zero e àquele neurônio não será ativado, dessa forma, pode-se afirmar que apenas alguns neurônios são ativados, tornando a rede esparsa.

Dessa forma, sua função é definida como a parte positiva de seu argumento, e ela pode ser representada pela equação 5.

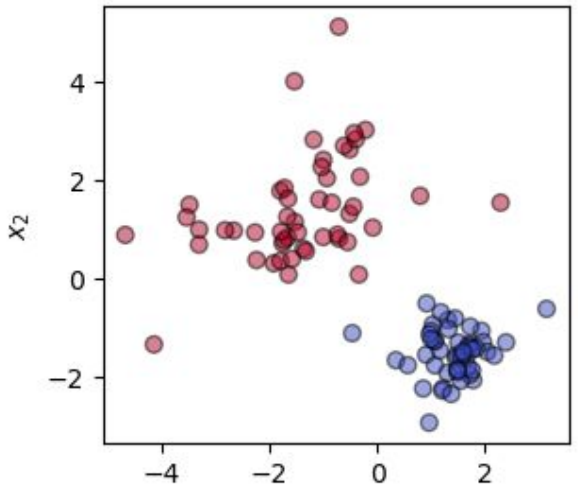
$$f(x) = x^+ = max(0, x) \quad (5)$$

2.1.2 Rede Neural Multicamadas

Quase quinze anos depois do modelo de neurônio proposto por McCulloch e Pitts [6], denominado como neurônio MCP, o psicólogo americano Frank Rosenblatt surgiu com o *Perceptron*, uma grande melhoria em relação ao modelo de neurônio MCP, pois ele conseguiu mostrar que os neurônios artificiais poderiam aprender com os dados. Rosenblatt criou um algoritmo de aprendizado supervisionado para o modelo de neurônio MCP modificado que permitia ao neurônio artificial descobrir os pesos corretos diretamente dos dados de treinamento por si só.

Apesar de revolucionário, o *Perceptron* idealizado por Rosenblatt é capaz apenas de realizar classificações binárias linearmente separáveis, ou seja, classifica os elementos de um determinado conjunto em dois grupos com base em uma regra prescrita. A Figura 3 descreve um problema de classificação binária linearmente separável.

Figura 3 - Problema de classificação binária linearmente separável.



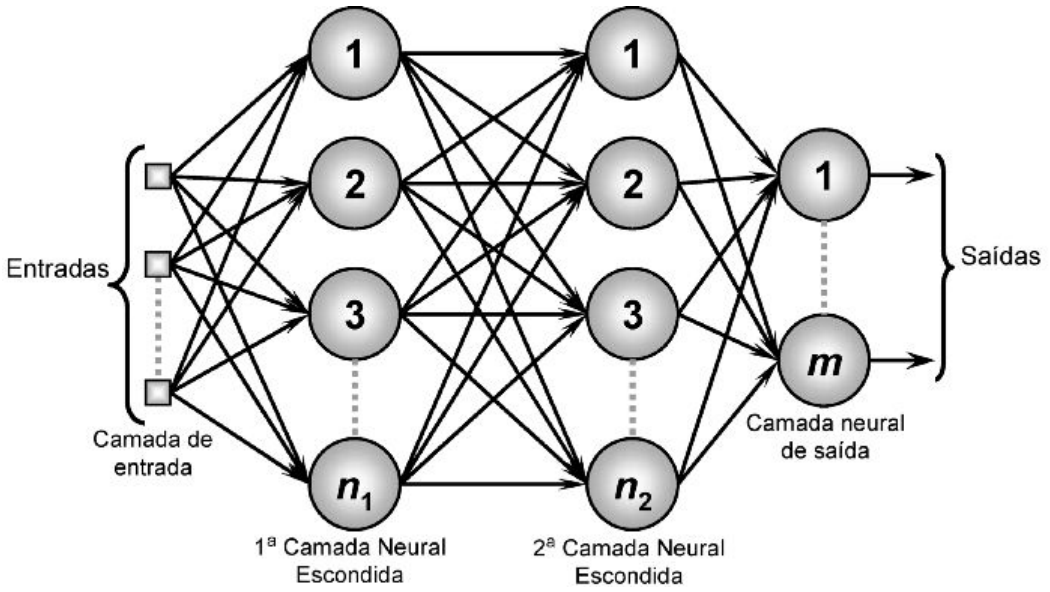
Fonte: Medium.com[1].

Dado que a maior parte dos problemas existentes não são linearmente separáveis, foi desenvolvida uma arquitetura mais robusta, o Perceptron Multicamadas (*Multilayer Perceptron*, MLP). O MLP é uma classe de rede neural artificial denominada de *feedfoward*, que consiste em uma rede em que as conexões entre os nós não formam ciclos [12].

Uma arquitetura MLP consiste em pelo menos três camadas de nós, a camada de entrada, a camada de saída e pelo menos uma camada oculta. Com exceção dos nós de entradas, cada nó é um neurônio e cada neurônio tem uma função de ativação não linear, como vimos na Subseção 2.1.1. O MLP utiliza uma técnica de aprendizado supervisionado conhecida como *backpropagation* para o aprendizado [13][14], que é um algoritmo utilizado para o reconhecimento de padrões e que resolve problemas não-linearmente separáveis. A Figura 4 ilustra um modelo de Perceptron Multicamadas genérico.

[1] Disponível em: https://medium.com/@b.terryjack/deep-learning-background-research-64578f0d551d

Figura 4 - Perceptron Multicamadas Genérico.



Fonte: Adaptado de Medium.com[2].

Na Figura 4 é possível ver com mais clareza a distribuição de camadas explicada anteriormente. Na primeira camada, nesse exemplo tem três neurônios, denominada de camada de entrada, são inseridos os atributos de entrada, as duas camadas conseguintes são as camadas ocultas e a última camada, que pode ter *m* neurônios, é a camada de saída, onde é atribuída a classificação do objeto de entrada. Como pode-se observar, o MLP se refere a redes totalmente conectadas, ou seja, cada neurônio de uma camada está conectado a todos os neurônios da próxima camada, o que as torna propensas a super ajustar os dados.

2.1.3 Função Softmax

A saída das redes neurais retornam valores arbitrários, onde o maior valor representa a classe "vencedora". A função *Softmax* gera um vetor que representa as distribuições de confiança de cada classe.

Ela é habitualmente utilizada como classificador para a camada de saída e tem como objetivo transformar esses números arbitrários em probabilidade que somam 1 [15].

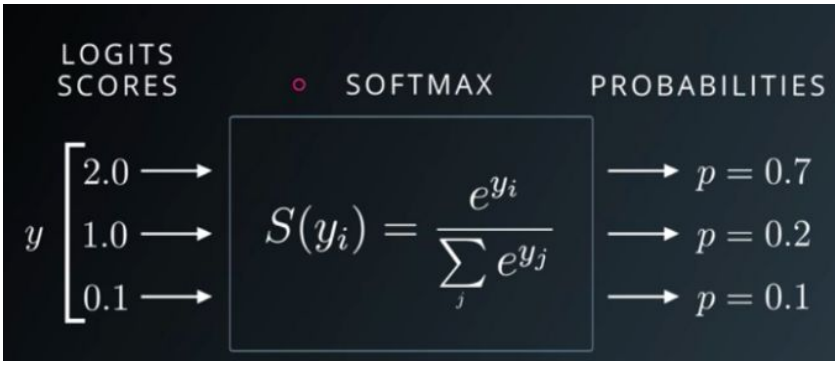
Na equação 6 é possível perceber que a função *Softmax* gera valores apenas entre 0 e 1, como dito anteriormente.

[2] Disponível em: https://medium.com/ensina-ai/rede-neural-perceptron-multicamadas-f9de8471f1a9

$$Softmax(y_i) = \frac{exp(y_i)}{\sum_j exp(y_j)} \quad (6)$$

A Figura 5 exemplifica o uso da função Softmax. No exemplo, a função transforma os valores de entrada [2.0, 1.0, 0.1] em probabilidades [0.7, 0.2, 0.1] e a soma dessas probabilidades é 1.

Figura 5 - Exemplo de Aplicação da Função Softmax.



Fonte: Sefiks.com[3].

2.1.4 Codificação One-Hot

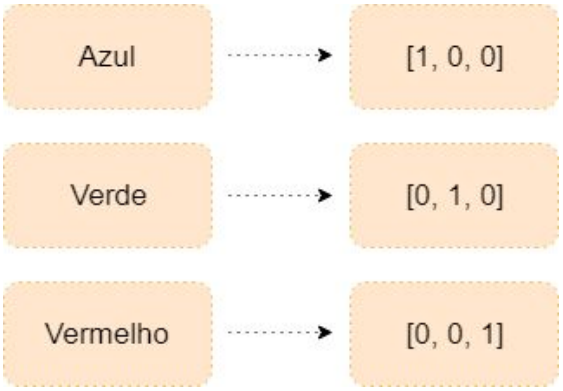
Alguns algoritmos de redes neurais artificiais não trabalham com dados categóricos, ou seja, não trabalham com dados do tipo nominal ou ordinal. Por causa disso, esses algoritmos não aceitam esse tipo de dado como entrada. Diante dessa limitação, precisamos converter as variáveis categóricas para valores numéricos.

Além disso, usa-se a codificação para dados de entrada inteiros também, para transformar o dado em matriz. Organizar o formato como matriz é uma forma de otimizar a leitura de rótulos durante o treinamento, por isso utiliza-se para dados categóricos e não categóricos.

Existem diversos algoritmos que realizam essa codificação de valores categóricos para valores números. O algoritmo utilizado neste trabalho é a codificação One-Hot, que gera vetores binários para cada valor inteiro, dessa forma, cada valor inteiro agora é representado por um vetor binário único. A Figura 6 exemplifica a codificação one-hot.

[3] Disponível em: https://sefiks.com/2017/11/08/softmax-as-a-neural-networks-activation-function/.

Figura 6 - Exemplo de um codificador One-hot.



Fonte: Elaborado pelo autor.

2.1.5 Função de Custo

Normalmente, com redes neurais, procuramos minimizar o erro. Como tal, a função objetiva é muitas vezes referida como uma função de custo ou uma função de perda e o valor calculado pela função de perda é referido simplesmente como "perda" (do inglês, *loss*) .

A função de custo ou perda tem um trabalho importante, pois deve destilar fielmente todos os aspectos do modelo para um único número, de modo que as melhorias nesse número sejam um sinal de um modelo melhor.

Ao calcular o erro do modelo durante o processo de otimização, uma função de perda deve ser escolhida.

2.1.6 Cross Entropy

A perda de *Cross-Entropy* mede o desempenho de um modelo de classificação cuja saída é um valor de probabilidade entre 0 e 1. A perda aumenta à medida que a probabilidade prevista diverge do rótulo real, ou seja, quanto menor for o valor da perda, melhor é a precisão da rede.

Portanto, prever uma probabilidade de 0,012 quando a etiqueta de observação real é 1 seria ruim e resultaria em um alto valor de perda. Um modelo perfeito teria uma perda de 0.

2.1.7 Treinamento de uma Rede Neural Multicamadas

No treinamento de uma rede neural multicamadas, os pesos sinápticos e o *bias* são ajustados a cada iteração de forma que o vetor de saída se aproxime do valor esperado para a saída. Esse ajuste é realizado através do algoritmo de retropropagação de erro, que consiste em duas etapas, a propagação e a retropropagação (forward-propagation e backpropagation, respectivamente) [16].

Na propagação, uma entrada é aplicada ao à rede neural e o seu resultado se propaga pela rede, em todas as camadas da rede, passando pela camada de entrada, as camadas ocultas e finalizando na camada de saída. Na última camada é comum utilizar a função *Softmax* para produzir uma saída adequada para ser comparada com o rótulo gerado pela codificação one-hot. Nessa etapa, os pesos da rede permanecem inalterados.

Ao finalizar a etapa da propagação, após a rede gerar os valores estimados por ela, é definida a função de perda. Com a função de perda é estabelecido o desempenho dessa rede e se o desempenho não for suficiente, é iniciada a fase de retropropagação.

Na retropropagação, um sinal de erro é calculado no fim da rede e esse sinal é retro propagado por todas as camadas da rede no sentido reverso e, ao fim desse processo, os pesos da rede são ajustados. Para isto, é necessário calcular a função *cross entropy* e a função de perda e utilizar o método do gradiente descendente.

A cada ciclo os pesos são ajustados com base na quantidade de erros na saída em comparação com o resultado esperado.

2.2 Aprendizado Profundo

O Aprendizado Profundo (DL) é um subconjunto de ML, no qual, as RNAs se adaptam e aprendem a partir de vastas quantidades de dados, utilizando múltiplas camadas

para extrair progressivamente características a partir de uma entrada bruta, a fim de aprender a prever e classificar informações [17].

A maioria das arquiteturas modernas de aprendizagem profunda é baseada em RNAs e usa várias camadas de unidades de processamento não-linear para extração e transformação de características. Cada camada sucessiva usa a saída da camada anterior para sua entrada. O 'profundo', em aprendizado profundo, remete ao número de camadas pelo qual os dados são processados, ou seja, mais de duas camadas ocultas.

A motivação do desenvolvimento de algoritmos de DL foi, em grande parte, pela ineficiência dos algoritmos tradicionais em solucionar problemas de IA. Existem muitos fatores que diferenciam as técnicas de DL das técnicas clássicas de machine learning, e alguns desses fatores favorecem a utilização dessas técnicas em áreas como visão computacional e processamento de linguagem natural [15].

Essa técnica só se tornou possível devido à grande quantidade de dados disponíveis nos dias atuais. Em 2015, entramos na era do zetabytes, e atualmente geramos mais de 2,5 quintilhões de bytes diariamente [18]. Tendo em vista a quantidade de dados e considerando o crescimento de parâmetros a serem ajustados no algoritmo de aprendizado, por causa do aumento de camadas de uma RNA, é necessário uma alta capacidade de processamento do computador.

2.2.1 Função de Otimização

Hoje, todas as bibliotecas de DL de última geração, como Caffe [19]e Keras [20], contém várias implementações de vários algoritmos de otimização (por exemplo, método do gradiente descendente, Adadelta e Adam). Esses algoritmos, no entanto, são frequentemente usados como otimizadores de caixa preta, pois é difícil encontrar explicações práticas de seus pontos fortes e fracos.

2.2.1.1 Adadelta

A função Adadelta é uma extensão do Adagrad sendo uma versão mais robusta. É um algoritmo para otimização baseados no gradiente descendente, porém, ele adapta a taxa de aprendizado aos parâmetros, realizando atualizações maiores para parâmetros infreqüentes e menores para parâmetros frequentes. Por esse motivo, é adequado para lidar com dados esparsos [21].

2.2.2 Normalização de Batch

A normalização de Batch é uma técnica proposta por Ioffe e Szegedy para melhorar o desempenho e a estabilidade de uma RNA [22]. Essa técnica busca normalizar as entradas de cada camada de forma que elas tenham uma ativação de saída média igual a 0 e desvio padrão igual a 1.

Normalizar as entradas de uma rede ajuda a aprender, ou seja, considerando que uma rede é apenas uma série de camadas, onde a saída de uma camada se torna a entrada para a próxima camada, podemos pensar em qualquer camada em uma rede neural como a primeira camada de uma rede subseqüente menor. Dessa forma, pensando como uma série de redes que se alimentam, aplica-se a normalização à saída de cada camada antes da função de ativação.

2.3 Rede Neural Convolucional

A Rede Neural Convolucional (*Convolutional Neural Networks*, CNN) pode ser considerada uma variante da rede neural MLP que subdivide um dado de entrada para tentar extrair características de cada conjunto. As CNNs podem captar uma entrada, atribuir importância, como pesos e *bias*, a vários aspectos dessa entrada e ser capaz de classificá-la [23].

Elas são muito utilizadas em aplicações de reconhecimento de imagem e vídeo, sistema de recomendação classificação de imagem, análise de imagens médicas e processamento de linguagem natural.

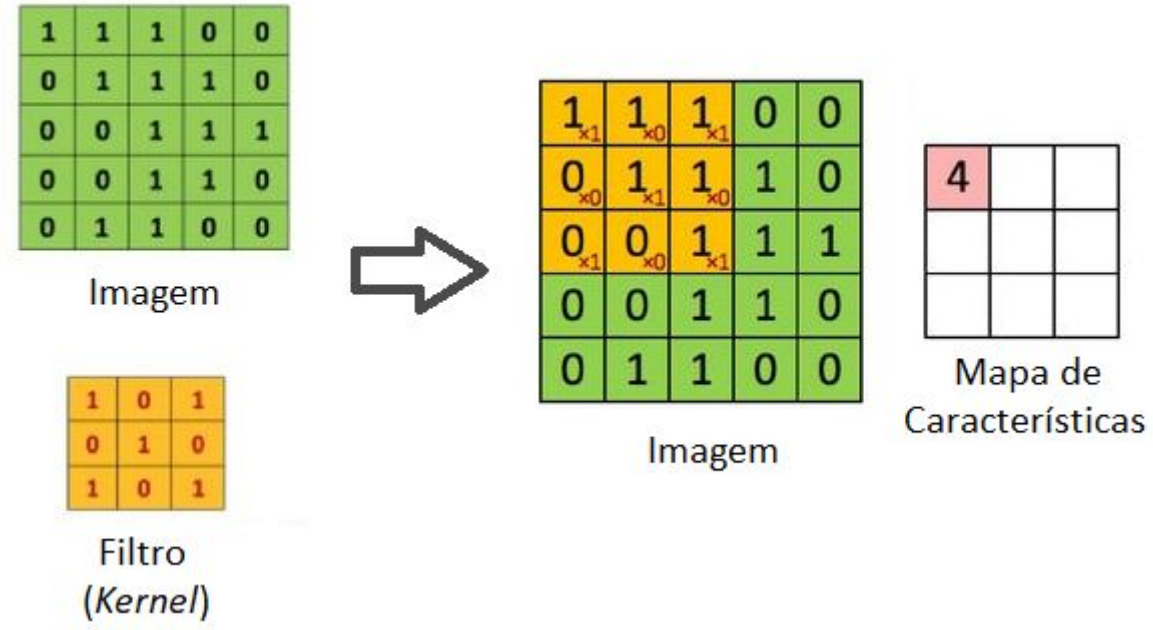
Em vez de usar camadas ocultas totalmente conectadas, como a MLP, a arquitetura de uma CNN se baseia na alternação de camadas convolucionais e operações de *pooling*. Cada camada possui um conjunto de filtros, também conhecido como *kernel*, que são responsáveis por extrair características locais de uma entrada. Com treinamento suficiente, a rede aprende esses filtros que nos algoritmos tradicionais são projetados à mão. Com isso, mapas de convolução podem ser criados, contendo características específicas [24].

2.3.1 Camada de Convolução

Uma camada convolucional é formada por um conjunto de neurônios, que se conectam a subregiões de imagens, podendo ser de entrada ou saída de uma camada anterior, onde é aplicado um conjunto de filtros convolucionais que percorrem toda a imagem e são responsáveis por extrair mapas de características (*feature maps*). Cada filtro possui um conjunto de pesos que são ajustados automaticamente, dessa forma, diferentes filtros aprendem tipos de características diferentes da entrada, como orientação de bordas, intensidade de cor, contornos e formas [24].

A Figura 7 é um exemplo de uma operação de convolução, onde a entrada 5x5 é varrida a cada passo por um filtro de tamanho 3, ou seja, com um *stride* de 1, sendo o *stride* uma ferramenta que indica como o filtro avança sobre a entrada, podendo ser de 1 em 1 ou 2 em 2 passos, etc. [25].

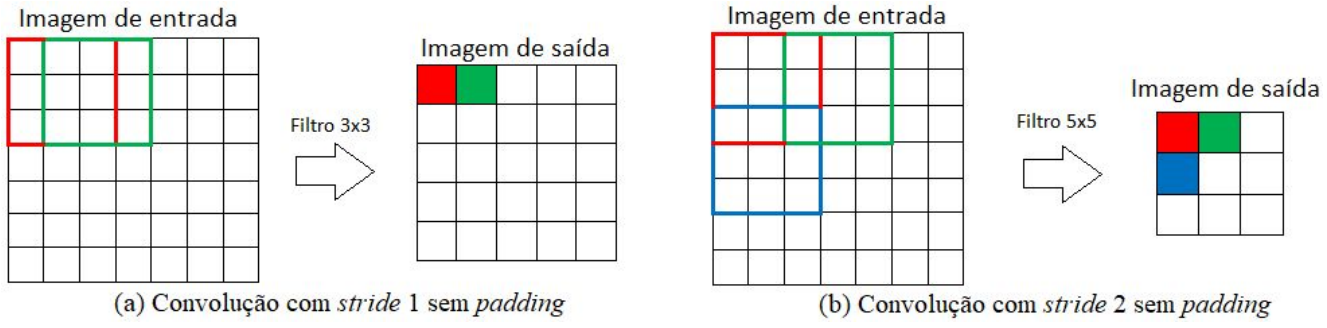
Figura 7 - Operação de Convolução.



Fonte: Adaptado de http://deeplearning.stanford.edu/.

Outra ferramenta importante utilizada na camada de convolução é o preenchimento (*padding*), que consistem adicionar bordas ao redor da imagem para que a imagem não perca tamanho a cada camada convolucional, *pooling* e *strides*, ou seja, quando essas operações são utilizadas sem o *padding*, a imagem vai sempre perdendo dimensionalidade, como pode-se observar na Figura 8.

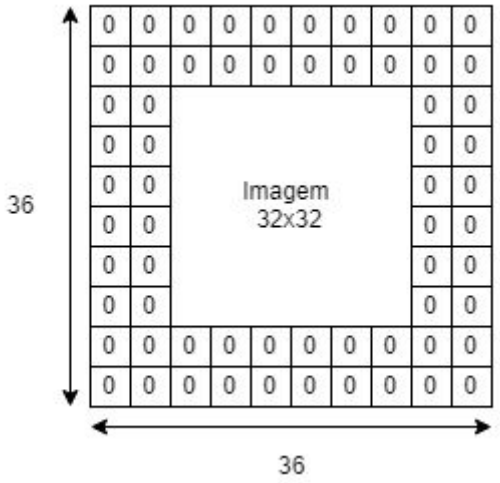
Figura 8 - Exemplo de convolução com uso de *strides*.



Fonte: Elaborado pelo autor.

O *padding* possibilita também a extração de características que estão contidas próximas às bordas. A Figura 9 exemplifica como é feito o preenchimento em uma imagem de tamanho 32x32 e filtro 5x5.

Figura 9 - Exemplo de utilização de *padding*.



Fonte: Elaborado pelo autor.

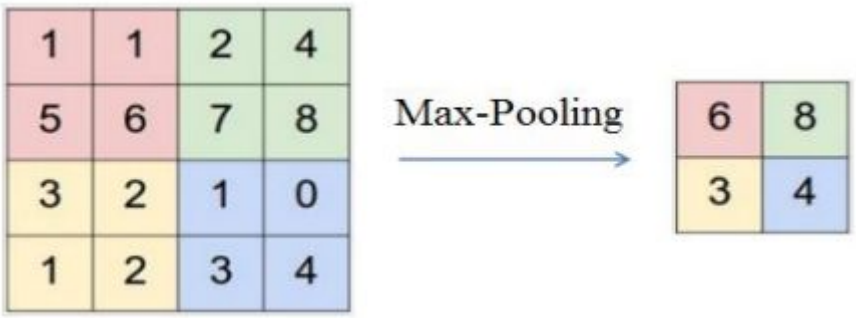
2.3.4 Camada de *Pooling*

A camada de *Pooling*, também conhecida como subamostragem, é comumente utilizada após a camada de convolução. A camada de convolução gera mapas de características e a camada de *pooling* opera em cada mapa separadamente, reduzindo as suas dimensionalidades e fornece um novo mapa de características, sendo uma espécie de versão reduzida de sua entrada [26]. A importância dessa redução é para diminuir o custo computacional da rede e para evitar o super ajustamento.

Existem várias formas de subamostragem aplicáveis a um mapa de característica, porém, apenas duas são mais utilizadas, são elas selecionar o valor máximo (*max pooling*) e selecionar a média (*average pooling*) em cada trecho de cada mapa.

A Figura 10 demonstra um mapa de característica e o seu resultado após aplicar uma operação de *max pooling* com filtros de tamanho 2x2 e *stride* igual a 2, resultando em uma imagem reduzida pela metade.

Figura 10 - *Max Pooling* com um filtro de tamanho 2x2 e *stride* 2.



Fonte: HIJAZI, Samer.

O *max pooling*, ao pegar o maior valor, elimina valores desprezíveis ou ruídos, criando uma invariância a pequenas mudanças e distorções locais [27]. Esta foi a técnica utilizada neste trabalho.

2.3.5 Camada Totalmente Conectada (*Fully Connected Layer*)

As camadas totalmente conectadas (do inglês, *Fully Connected Layer* - FCL) são semelhantes às MLPs convencionais, pois conecta todos os neurônios da camada anterior a ela com todos os neurônios dessa camada. A FCL combina todas as características aprendidas nas camadas anteriores, camadas convolucionais, tornando possível a classificação feita na camada de saída, normalmente utilizando a função *softmax*, que consiste na confiança da rede.

3 METODOLOGIA

Neste capítulo serão descritos o material e métodos utilizados para a solução do problema apresentado. A metodologia consiste no treinamento e validação de uma rede neural convolucional para aprendizado de classificação, detecção e contagem de faces em ambientes não controlados a partir de uma base de dados própria com imagens de salas de aula de uma autoescola. Para isso o experimento foi dividido em três etapas, pré-processamento da base de dados, treinamento da CNN para a classificação e a detecção das faces a partir da rede treinada.

3.1 Classificação

3.1.1 Material

A rede neural artificial e o pré-processamento dos dados foram implementados em Python 3.7.3, utilizando as bibliotecas Keras e Numpy. Todos os códigos foram feitos e rodados em um computador com CPU Intel® Core™ i7-7500 2.7 GHz, GPU NVIDIA GeForce® 940MX 4GB VRAM e memória RAM de 16GB DDR4.

3.1.1.1 Keras

A biblioteca escolhida para a implementação deste trabalho foi Keras pela sua simplicidade para desenvolver modelos. Keras é uma biblioteca de rede neural de alto nível de código aberto, escrita em Python e capaz de rodar sobre TensorFlow, Microsoft Cognitive Toolkit, Theano ou PlaidML, bibliotecas de código aberto para aprendizado de máquina aplicáveis a uma ampla variedade de tarefas [20].

3.1.1.2 Base de Dados

No presente trabalho foi utilizada apenas uma base de dados sigilosa e, portanto, as imagens não poderão ser divulgadas, portanto, a análise dos resultados será apenas textual.

Esta base contém imagens de salas de aula, totalizando 8919 imagens, com a localização das faces das pessoas de cada imagem documentada em um arquivo *.xml*.

Como o método proposto utiliza uma abordagem de técnica de aprendizagem supervisionada, foi necessária a criação de uma nova base de dados com duas classes intituladas de 'face' e nao_face'.

Para popular a classe 'nao_face' foi desenvolvido um algoritmo que seleciona partes aleatórias das imagens de sala de aula que não tenham interseção com as faces gabaritadas.

Após a extração das imagens, a base ficou com uma totalidade de 259.201 imagens, sendo 124.690 imagens pertencentes à classe 'face' e 134.511 imagens pertencentes à classe 'nao_face'.

3.1.2 Métricas

Com o intuito de avaliar a performance do método proposto, foram utilizadas algumas métricas bem comum na literatura de aprendizagem profunda. Essas métricas abrangem medidas chave para avaliar tarefas de classificação e detecção como: verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN).

A primeira métrica a ser avaliada é a acurácia (1), em problemas de classificação, a acurácia é medida de acordo com o número de predições corretas feita pelo modelo sobre toda as predições feitas.

$$Acurácia = \frac{tp+tn}{tp+fp+tn+fn} \quad (1)$$

Utilizar apenas a acurácia não é suficiente para avaliar o modelo, portanto, foi utilizada uma segunda métrica para complementá-la, a matriz de confusão.

A matriz de confusão (2), é uma medida de desempenho para o problema de classificação de aprendizado de máquina, nesse trabalho em questão, a saída são duas classes. É uma Tabela com 4 combinações diferentes de valores previstos e reais.

Tabela 1 - Exemplificação de uma matriz de confusão.

		Valor Real	
		Positivo	Negativo
Valor Previsto	Positivo	TP	FP
	Negativo	FN	TN

3.1.3 Pré Processamento

3.1.3.1 Imagem

As faces extraídas das imagens originais têm tamanhos variados, para o tratamento dessas imagens foi utilizada uma biblioteca de manipulação de imagens em python (do inglês *Python Imaging Library* - PIL) para redimensionar as imagens utilizando a mesma proporção para largura e altura, uma vez que, para o treinamento das redes, é necessário que todas as imagens possuam o mesmo tamanho.

3.1.3.2 Equalização de Histograma

Na base de dados há muitas imagens escuras e que tornam o seu conteúdo difícil de identificar. Com o intuito de melhorar a qualidade dessas imagens, foi aplicada uma técnica de equalização de histograma, a equalização do histograma adaptativo com contraste limitado (do inglês, Contrast-limited adaptive histogram equalization - CLAHE).

A CLAHE é uma técnica de processamento de imagem usada para melhorar o contraste de uma imagem. Esse método adaptativo calcula vários histogramas, cada um correspondendo a uma seção distinta da imagem, e os utiliza para redistribuir os valores de luminosidade da imagem. Portanto, é adequado para melhorar o contraste local e aprimorar as definições de bordas em cada região de uma imagem [28].

As imagens foram convertidas para o modelo de cor Lab, espaço de cores definido por Richard S. Hunter em 1948 [29] [30], onde o canal ‘L’ representa a luminosidade da imagem (*lightness*) e os canais ‘a’ e ‘b’ representam as cores (verde-vermelho e azul-amarelo, respectivamente). Após a conversão, o método CLAHE foi aplicado no canal de luminosidade (L) da imagem e, por fim, as imagens foram convertidas para o formato tradicional RGB.

3.1.3.3 Codificação One-hot

Conforme explicado anteriormente na Subseção 2.1.4, RNAs não podem operar com rótulos diretamente. Tendo em vista essa limitação, foi desenvolvida a codificação one-hot dos rótulos das classes de objetos. Na primeira coluna temos os rótulos de cada classe e em seguida as linhas que formam os vetores que irão representá-los.

Tabela 2 - Codificação One-hot das classes.

face	1	0
nao_face	0	1

3.1.4 Treinamento da rede

3.1.4.1 Conjunto de Dados de Treinamento, Validação e Teste

A base foi dividida em três partes denominadas de treinamento, validação e teste. A amostra de treinamento é usada para treinar o algoritmo e os dados restantes a amostra de validação são usados para avaliar o desempenho do algoritmo.

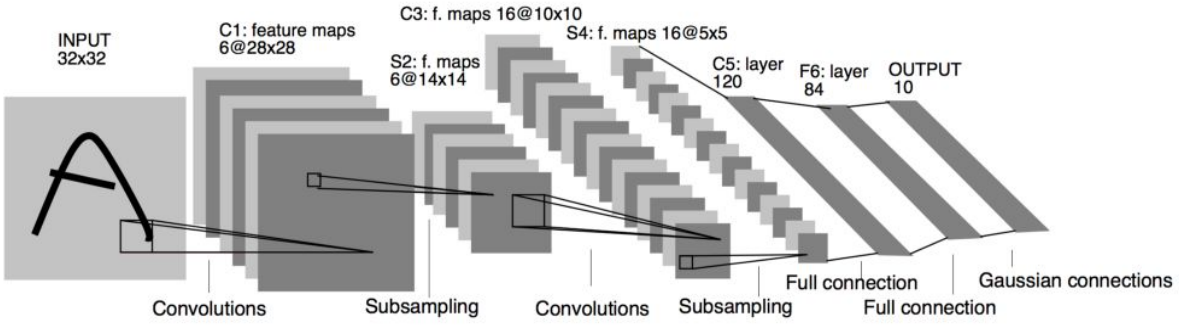
Não há um modelo de divisão considerado mais adequado, a adequação varia de acordo com o problema abordado e com a base de dados com que se está trabalhando. Considerando a quantidade de imagens na base de dados em questão, a divisão foi feita na seguinte proporção e de forma aleatória: 90% das imagens para o treinamento, totalizando 233.280 imagens, 5% para a validação e os 5% restantes para o teste, totalizando 12.960 e 12.961 imagens para cada etapa, respectivamente.

Isso significa que, durante o processo de treinamento, utiliza-se as imagens de validação para verificar como o algoritmo desempenha a tarefa de identificação em imagens com as quais ele não foi treinado. E ao final de todo o processo de treinamento é realizado uma nova verificação no conjunto de teste, que é de onde provêm os dados informados na Seção de resultados.

3.1.4.2 Arquitetura

A arquitetura implementada neste trabalho foi baseada na LeNet-5, método proposto por LeCun, utilizado em grande escala para classificar automaticamente dígitos escritos à mão em cheques bancários nos Estados Unidos [31]. Ela foi escolhida por ser uma rede enxuta, com poucas camadas convolucionais e poucos filtros. A arquitetura original da LeNet-5 é composta por sete camadas, sendo três camadas convolucionais (C1, C3, C5), duas camadas de sub-amostragem (S2 e S4) e duas camadas densas ou totalmente conectada (F6, Output), que são seguidas por uma camada de saída, e a sigmóide como função de ativação. O *padding* utilizado foi o *valid*, que significa que não há preenchimento. As imagens de entrada em tamanho 32x32 pixels (32 de largura, 32 de altura e 3 canais de cor).

Figura 11 - Arquitetura original da LeNet-5.



Fonte: LECUN, Yann.

A rede desenvolvida, tendo como base a LeNet-5, possui três camadas convolucionais, duas camadas de sub-amostragem e três camadas totalmente conectadas e, assim como a LeNet-5, as imagens de entrada possuem dimensão 32x32x3. A Tabela 2 apresenta um sumário breve da arquitetura proposta. Foram realizados testes com as funções de ativação sigmóide, ReLu e tangente hiperbólica (*tanh*) com o intuito de encontrar uma rede que gerasse melhores resultados. Neste trabalho, foi utilizada a *Cross-Entropy*, que é mais indicada para problemas de classificação binária ou multiclasses.

Tabela 3 - Sumário da arquitetura da rede.

Camada	Tipo de Camada	Mapas de Ativação	Tamanho	Stride	Padding	Tamanho do Kernel
Entrada	Imagem	1	32x32			-
1	Convolução	6	28x28	1	valid	5x5
2	Max Pooling	6	14x14	2	valid	2x2
3	Convolução	16	10x10	1	valid	5x5
4	Max Pooling	16	5x5	2	valid	2x2
5	Convolução	120	1x1	1	valid	5x5
6	Totalmente Conectada	-	84			
7	Totalmente Conectada	-	48			
Output	Totalmente Conectada	-	2			-

3.2 Detecção

Para a detecção, foi utilizado um algoritmo cedido pelo aluno de mestrado Paulo Ricardo P. Silva. Para a detecção, a base de dados utilizada foi a original, que contém as imagens de salas de aula. O modelo utilizado para a detecção foi o que obteve melhor acurácia, que será mencionado posteriormente na subseção 4.1.

Neste algoritmo, as imagens das salas de aula são percorridas por uma janela de 70x70 pixels com *stride* 10 e, a cada passo, a janela extraída da imagem original é redimensionada para o tamanho de entrada do modelo, 32x32 pixels, e essa imagem é passada para o modelo treinado da rede para que o modelo retorne a confiança de que há ou não uma face naquela janela.

E, logo após toda a imagem ser percorrida, é feita uma filtragem para que o algoritmo exiba apenas as janelas em que o modelo tenha alcançado uma confiança acima de 90% de que há ou não a face.

4 RESULTADOS

Neste capítulo será apresentado os resultados referentes ao desempenho da arquitetura proposta para classificação e detecção de faces.

4.1 Classificação

Os resultados que serão apresentados foram obtidos com os parâmetros de inicialização descritos na Tabela 3 e método de subamostragem *max pooling*.

Tabela 4 - Parâmetros de inicialização.

Épocas	Taxa de Aprendizagem	Batch
10	1	32

Com a arquitetura proposta, foram testadas três funções de otimização, são elas Adadelta, Adam e o Método do Gradiente Descendente Estocástico (do inglês, *Stochastic Gradient Descent* - SGD). As funções Adadelta e Adam obtiveram resultados semelhantes e melhores que o SGD, porém o Adadelta alcançou uma acurácia minimamente superior ao Adam, uma média de 0,15 ponto percentual, portanto, os resultados que serão mostrados abaixo foram realizados com a Adadelta mantendo seus parâmetros padrões.

Como citado na Subseção 3.4.2, foram realizados testes com as funções de ativação ReLu, sigmóide e tangente hiperbólica (*tanh*). A Tabela 4 apresenta uma comparação da acurácia e perda para o teste da rede.

Tabela 5 - Desempenho da rede.

		Acurácia	Perda
Sem normalização de batch	ReLu	98.65%	4.16%
	Sigmóide	96.23%	5.53%
	Tanh	98.71%	4.44,%
Com normalização de batch	ReLu	98.21%	5.49%
	Sigmóide	96.11%	6.12%
	Tanh	98.36%	5.17%

A arquitetura original da LeNet-5 foi proposta com a função sigmóide e, de acordo com Xie [32], melhores resultados são gerados quando a função de ativação da arquitetura original é substituída pela ReLu. Neste trabalho foi possível constatar essa melhoria, pois, a substituição pela ReLu gerou um resultado consideravelmente melhor, com mais dois pontos percentuais de diferença.

De acordo com a literatura, no reconhecimento de imagens, a ReLu gera melhores resultados que a *tanh* [10], contudo, neste trabalho, não obtive esta confirmação pois a rede que utilizou a função de ativação *tanh* gerou o melhor resultado. Esse melhoria pode ser atribuída à base de dados utilizada neste trabalho.

A rede que não foi feita a normalização de *batch* e que foi utilizada a função de ativação *tanh* foi a que obteve o melhor resultado, apesar de serem resultados muito próximos. A diferença só foi considerável quando comparada à rede com a função de ativação original.

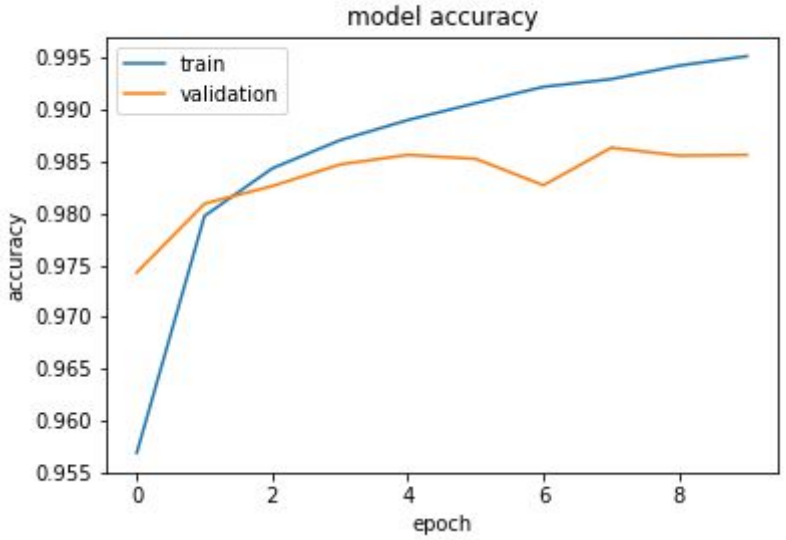
Na Tabela 5 é possível observar a matriz de confusão dos dados de teste, a Figura 12 apresenta a evolução da aprendizagem enquanto a perda vai diminuindo ao longo das épocas referente à validação e ao treinamento e a Tabela 6 apresenta os resultados referentes ao tempo de processamento para treinamento e teste da rede, é possível visualizar que a rede desenvolvida não necessita de muitos recursos computacionais, levando em consideração o

seu tempo de processamento. Os resultados apresentados são da rede que obteve melhor resultado.

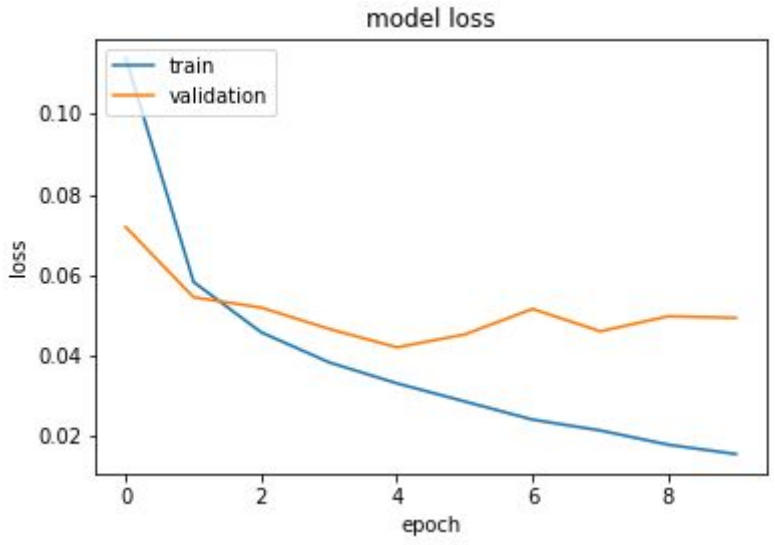
Tabela 6 - Matriz de confusão referente à classificação.

		Valor Real	
		Face	Não Face
Valor Previsto	Face	6652	93
	Não Face	74	6142

Figura 12 - Desempenho da rede.



(a) Acurácia.



(b) Perda.

Tabela 7 - Tempo de treinamento e teste da rede.

Etapa	Tempo (segundos)
Treinamento	3458
Teste	0,000452

Fazendo uma análise das imagens que a rede classificou de forma errada, foi possível perceber que há erro de gabarito. Apesar da rede ter classificado 93 falsos positivos, 15 dessas imagens estavam com o gabarito errado e realmente eram faces. Não considerando os erros de gabarito, a acurácia da rede subiria para 98.82%.

Ademais, levando em consideração os falsos negativos, algumas faces são de difícil reconhecimento para a rede, seja por iluminação, oclusão, ou mau posicionamento das faces, como por exemplo uma pessoa de perfil ou com a cabeça abaixada. Esse erro pode ter sido ocasionado porque a rede não treinou com muitas imagens com estes ângulos ou iluminação.

Por fim, algumas imagens, tanto falso negativo quanto falso positivo, não há justificativa aparente para o erro, são imagens sem problema de oclusão, iluminação, pose ou posicionamento de câmera ruim.

A fim de melhorar a acurácia desta rede, o método CLAHE foi aplicado primeiramente nas imagens que a rede errou na classificação com o intuito de testá-la novamente na mesma rede só que com essas imagens processadas. Após aplicar o CLAHE a rede passou a acertar 88 das 167 imagens que havia errado na classificação, cerca de 52% de acerto.

Por fim, o método CLAHE foi aplicado em todas as imagens da base de dados e a rede foi treinada novamente com essas imagens pré-processadas, a arquitetura utilizada foi a que obteve o melhor resultado e com os mesmos parâmetros apresentados acima.

A acurácia alcançada pela rede após a aplicação do CLAHE nas imagens foi de 98.45% e teve uma perda de 5.91%. Se manteve inferior, porém, próximo à acurácia da rede com as imagens originais.

Não houve uma melhoria no resultado e isso se deve porque a maioria das imagens da base não precisam desse pré processamento e seria necessário fazer uma filtragem para aplicar esse método apenas nas imagens que necessitam dessa melhoria no contraste.

4.2 Detecção

Foram realizados teste com todas as redes apresentadas na Subseção 4.1 mas os melhores resultados, e os que serão expostos, são referentes à rede que obteve a melhor acurácia. Não foi utilizada nenhuma métrica de desempenho para avaliar o modelo portanto, foi feita apenas uma avaliação visual.

Observa-se que a rede não foi capaz de detectar todas as pessoas da imagem, além de ter feito detecções erradas. Esperava-se que ela fosse capaz de realizar a detecção com um pouco mais de eficiência, considerando que a acurácia média da classificação foi 98%. Não foi possível realizar os devidos ajustes no algoritmo por falta de tempo hábil.

5 CONCLUSÃO E TRABALHOS FUTUROS

Para este trabalho, me propus a fazer uma rede convolucional para a realizar a classificação e detecção de pessoas em ambientes não controlados e consegui efetuar parcialmente a proposta do trabalho.

O método utilizado para a classificação gerou resultados bastante satisfatórios no ponto de vista da taxa de acurácia, mesmo considerando as dificuldades inerente à base utilizada, que tem muitos problemas de oclusão da face e iluminação.

Tendo em vista que os resultados obtido pela rede foram satisfatórios para uma rede básica, a princípio não foi necessário o estudo e implementação de arquiteturas mais avançadas.

Porém, apesar da detecção ter utilizado a rede que obteve a melhor acurácia na classificação, não obteve resultados visualmente satisfatórios. Esperava-se que a detecção gerasse bons resultados, pois a rede utilizada alcançou uma acurácia alta. Estima-se que esses erros ocorreram também por causa da escala das imagens e também por erros de lógica no algoritmo de detecção.

Por falta de tempo hábil, não foi possível avaliar o motivo desses erros. Como trabalho futuro pode ser feita também uma avaliação do algoritmo ter gerado resultados visualmente não satisfatórios e, também, podem ser definidas métricas de desempenho para o algoritmo de detecção.

Ainda como trabalho futuro, pode ser estudado e implementado uma rede neural convolucional mais complexa e pode ser desenvolvido um algoritmo que realize o reconhecimento de pessoas.

REFERÊNCIAS

[1] CHAN, Antoni B.; VASCONCELOS, Nuno. Counting People With Low-Level Features and Bayesian Regression. **IEEE Transactions on Image Processing**, vol. 21, p. 2160-2177, Abril 2012.

[2] XIONG, Guogang. et al.. An energy model approach to people counting for abnormal crowd behavior detection. vol. 83, p. 121-135, 2012.

[3] HOU, Ya-Li; PANG, Grantham K. H. Pang. People Counting and Human Detection in a Challenging Situation. **IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans**, vol. 41, no. 1, p. 24-33, Janeiro 2011.

[4] ZENG, Chenobin; MA, Huadong. Robust Head-Shoulder Detection by PCA-Based Multilevel HOG-LBP Detector for People Counting. **20th International Conference on Pattern Recognition**, p. 2069-2072, 2010.

[5] BISHOP, Christopher. **Pattern Recognition and Machine Learning**. 2.ed. Nova Iorque. Spring. 2006.

[6] McCulloch, Warren. S.; Pitts, Walter. A logical calculus of the ideas immanent in nervous activity. **Bulletin of Mathematics and Biophysics**, vol. 5, pp. 115-133, 1943.

[7] Hebb, Donald. The Organization of Behavior: A Neuropsychological Theory. New York. 1949.

[8] J. A. Hertz, A. S. Krogh, and R. G. Palmer, Introduction to the Theory of Neural Computation. Basic Books, 1.ed. vol. 1. 1991.

[9] HAHNLOSER, Richard. et al.. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature. 2000.

[10] GLOROT, Xavier; BORDES, Antoine; BENGIO, Yoshua (2011). Deep sparse rectifier neural networks. **Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics.** 2018.

[11] ROSENBLATT, Frank. **The perceptron**: a probabilistic model for information storage and organization in the brain. Psychological Review. 65(6), 386-408. 1958.

[12] ZELL, Andreas. Simulation of Neural Networks 1.ed. Addison-Wesley. p. 73. 1994.

[13] RUMELHART, David E.; Hinton, GEOFFREY E.; WILLIAMS, R. J.. Learning Internal Representations by Error Propagation. **Parallel distributed processing: Explorations in the microstructure of cognition**. vol. 1. Foundation. MIT Press. 1986.

[14] ROSENBLATT, Frank. **Principles of Neurodynamics**: Perceptrons and the Theory of Brain Mechanisms. Spartan Books. Washington DC. 1961.

[15] GOODFELLOW, Ian. et al. **Deep Learning**. MIT Press. 2016.

[16] RUSSELL, Stuart. e NORVIG, Peter. **Artificial intelligence**: A modern approach. Artificial Intelligence. Prentice-Hall, Englewood Cliffs. 1995.

[17] DENG, Li. YU, Dong. Deep Learning: Methods and Applications. Foundations and Trends® in Signal Processing: Vol. 7: No. 3–4, pp 197-387. 2014.

[18] ARTHUR, Charles. What's a zettabyte? By 2015, the internet will know, says Cisco. Disponível em: https://www.theguardian.com/technology/blog/2011/jun/29/zettabyte-data-internet-cisco. Acesso em: 14 de Agosto de 2019.

[19] Caffe. Disponível em: https://caffe.berkeleyvision.org. Acesso em: 14 de Agosto de 2019.

[20] Keras. Keras Documentation. Disponível em: https://keras.io. Acesso em: 14 de Agosto de 2019.

[21] RUDER, Sebastian. An overview of gradient descent optimization algorithms. 2016. Disponível em: http://ruder.io/optimizing-gradient-descent. Acesso em: 18 de Agosto de 2019.

[22] IOFFE, Sergey. SZEGEDY, Christian. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167. 2015.

[23] SAHA, Sumit. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. 2018. Disponível em: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. Acesso em: 22 de Agosto de 2019.

[24] VARGAS, Ana Caroline Gomes; PAES, Aline; VASCONCELOS, Cristina Nader. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. Proceedings of the XXIX Conference on Graphics. Patterns and Images. p. 1-4. 2016.

[25] LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. Nature. v. 521. n. 7553. p. 436. 2015.

[26] HIJAZI, Samer; KUMAR, Rishi; ROWEN, Chris. Using convolutional neural networks for image recognition.2015.

[27] ARAÚJO, Flávio H. D. et al.. Redes Neurais Convolucionais com Tensorflow: Teoria e Prática. III Escola Regional de Informática do Piauí. Livro Anais - Artigos e Minicursos. vol. 1. p. 382-406. 2017.

[28] PIZER, Stephen. et al.. Adaptive Histogram Equalization and Its Variations. Computer Vision, Graphics, and Image Processing. p. 355-368. 1987.

[29] HUNTER, Richard Sewall. Photoelectric Color-Difference Meter. **Proceedings of the Winter Meeting of the Optical Society of America**. Journal of the Optical Society of America. p. 651-651. vol. 38. 1948.

[30] HUNTER, Richard Sewall. Proceedings of the Thirty-Third Annual Meeting of the Optical Society of America. **Proceedings of the Thirty-Third Annual Meeting of the Optical Society of America**. Journal of the Optical Society of America. p. 651-651. vol. 38. 1948.

[31] LECUN, Yann et al.. Gradient-Based Learning Applied to Document Recognition. **Proceedings of the IEEE**. vol: 86 , Issue: 11. 1998.

[32] XIE, Yiliang, JIN, Hongyuan, TSANG, Eric C. C.. Improving the lenet with batch normalization and online hard example mining for digits recognition. **2017 International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR)**. IEEE. 2017.